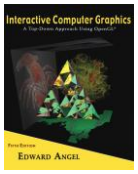


Name: _____ Term # ____

Homework 10 SOLUTIONS

(400 points)

NOTE: **Chapter 10 of the textbook shows the modeling and hierarchy in geometric objects.**
Part A is intended to be done by hand.
Part B is an OpenGL application.

A. **(300 pts) Paper and Pencil**

(Guidelines: *Read the material from the textbook chapter, you can use textbook figures to exemplify your answer, use keywords, summarize your answer, but the answer* ***cannot be longer the 7 lines!****)*

10.1 SYMBOLS and INSTANCES
a. Explain.
ANSWER:

10.2 HIERARCHICAL MODELS
a. Explain.
ANSWER:

10.3 A ROBOT ARM
a. Explain.
ANSWER:

10.4 TREES and TRAVERSAL
a. Explain.
ANSWER:

10.4.1 A Stack-Based Traversal
a. Explain.
ANSWER:

10.5 USE of TREE DATA STRUCTURES
Explain
ANSWER:

10.6 ANIMATION
a. Explain.
ANSWER:

10.7 GRAPHICAL OBJECTS
a. Explain.
ANSWER:

10.7.1 Methods, Attributes, and Messages
a. Explain.
ANSWER:

10.7.2 A Cube Object
a. Explain.
ANSWER:

10.7.3 Implementing the Cube Object
a. Explain.
ANSWER

10.7.4 Objects and Hierarchy
a. Explain.
ANSWER

10.7.5 Geometric Objects
a. Explain.
ANSWER

10.8 SCENE GRAPHS
a. Explain.
ANSWER:

10.9 A SIMPLE SCENE GRAPH API
a. Explain.
ANSWER:

10.9.1 The Node Class
a. Explain.
ANSWER

10.9.2 Geometry Nodes
a. Explain.
ANSWER

10.9.3 Camera Class
a. Explain.
ANSWER

10.9.4 Lights and Materials
a. Explain.
ANSWER

10.9.5 Transformations

a. Explain.
ANSWER

10.9.6 The Robot Figure
a. Explain.
ANSWER

10.9.7 Implementing the Viewer
a. Explain.
ANSWER

10.9.8 Implementing a Node
a. Explain.
ANSWER

10.10 OPEN SCENE GRAPH
a. Explain.
ANSWER:

10.11 GRAPHICS AND THE INTERNET
a. Explain.
ANSWER:

10.11.1 Networks and the Internet
a. Explain.
ANSWER:

10.11.2 Hypermedia and HTML
a. Explain.
ANSWER:

10.11.3 Databases and VRML
a. Explain.
ANSWER:

10.11.4 JAVA and Applets
a. Explain.
ANSWER:

10.12 OTHER TREE STRUCTURES

a. Explain.
ANSWER:

10.12.1 CSG Trees

a. Explain.
ANSWER:

10.12.2 BSP Trees

a. Explain.
ANSWER:

10.12.3 Quadtrees and Octrees

a. Explain.
ANSWER:

B. (50 pts) Visual Studio 2008 C++ Project

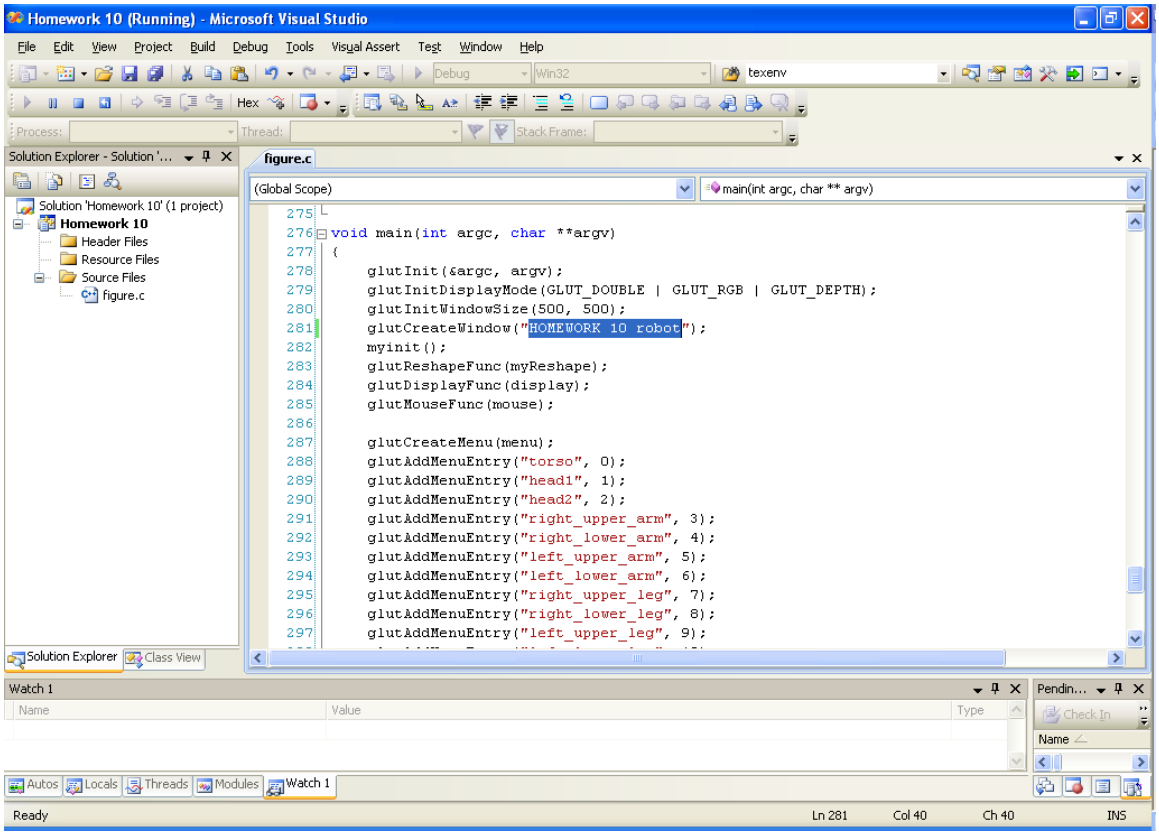
B1. Create Visual Studio 2008 C++, Empty Project, Homework10:

Modify the dynamic.c from Class Participation on Lecture 10:

1. `glutCreateWindow("HOMEWORK 10 robot");`
2. `The "robot" is made up with the glutSolidCube glut Object`
3. `"robot" tranversed directly in application, "pushing" and "poping" matrices and attributes as necessary.`

Build and run this Project: Insert a screenshot of your output.

ANSWER:




```
// figure.c

/* Interactive Figure Program from Chapter 8 using cylinders (quadrics)
*/
/* Style similar to robot program but here we must traverse tree to
display */
/* Cylinders are displayed as filled and light/material properties */
/* are set as in sphere approximation program */

#include <stdlib.h>
```

```
#ifdef __APPLE__
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif

#define TORSO_HEIGHT 5.0
#define UPPER_ARM_HEIGHT 3.0
#define LOWER_ARM_HEIGHT 2.0
#define UPPER_LEG_RADIUS  0.5
#define LOWER_LEG_RADIUS  0.5
#define LOWER_LEG_HEIGHT 2.0
#define UPPER_LEG_HEIGHT 3.0
#define UPPER_LEG_RADIUS  0.5
#define TORSO_RADIUS 1.0
#define UPPER_ARM_RADIUS  0.5
#define LOWER_ARM_RADIUS  0.5
#define HEAD_HEIGHT 1.5
#define HEAD_RADIUS 1.0

typedef float point[3];

static GLfloat theta[11] = {0.0,0.0,0.0,0.0,0.0,0.0,0.0,
            180.0,0.0,180.0,0.0}; /* initial joint angles */
static GLint angle = 2;

GLUquadricObj *t, *h, *lua, *lla, *rua, *rla, *lll, *rll, *rul, *lul;

double size=1.0;

void torso()
{
   glPushMatrix();
   glRotatef(-90.0, 1.0, 0.0, 0.0);
   gluCylinder(t,TORSO_RADIUS, TORSO_RADIUS, TORSO_HEIGHT,10,10);
   glPopMatrix();
}

void head()
{
   glPushMatrix();
   glTranslatef(0.0, 0.5*HEAD_HEIGHT,0.0);
   glScalef(HEAD_RADIUS, HEAD_HEIGHT, HEAD_RADIUS);
   gluSphere(h,1.0,10,10);
   glPopMatrix();
}

void left_upper_arm()
{
   glPushMatrix();
   glRotatef(-90.0, 1.0, 0.0, 0.0);
   gluCylinder(lua,UPPER_ARM_RADIUS, UPPER_ARM_RADIUS,
UPPER_ARM_HEIGHT,10,10);
   glPopMatrix();
}
```

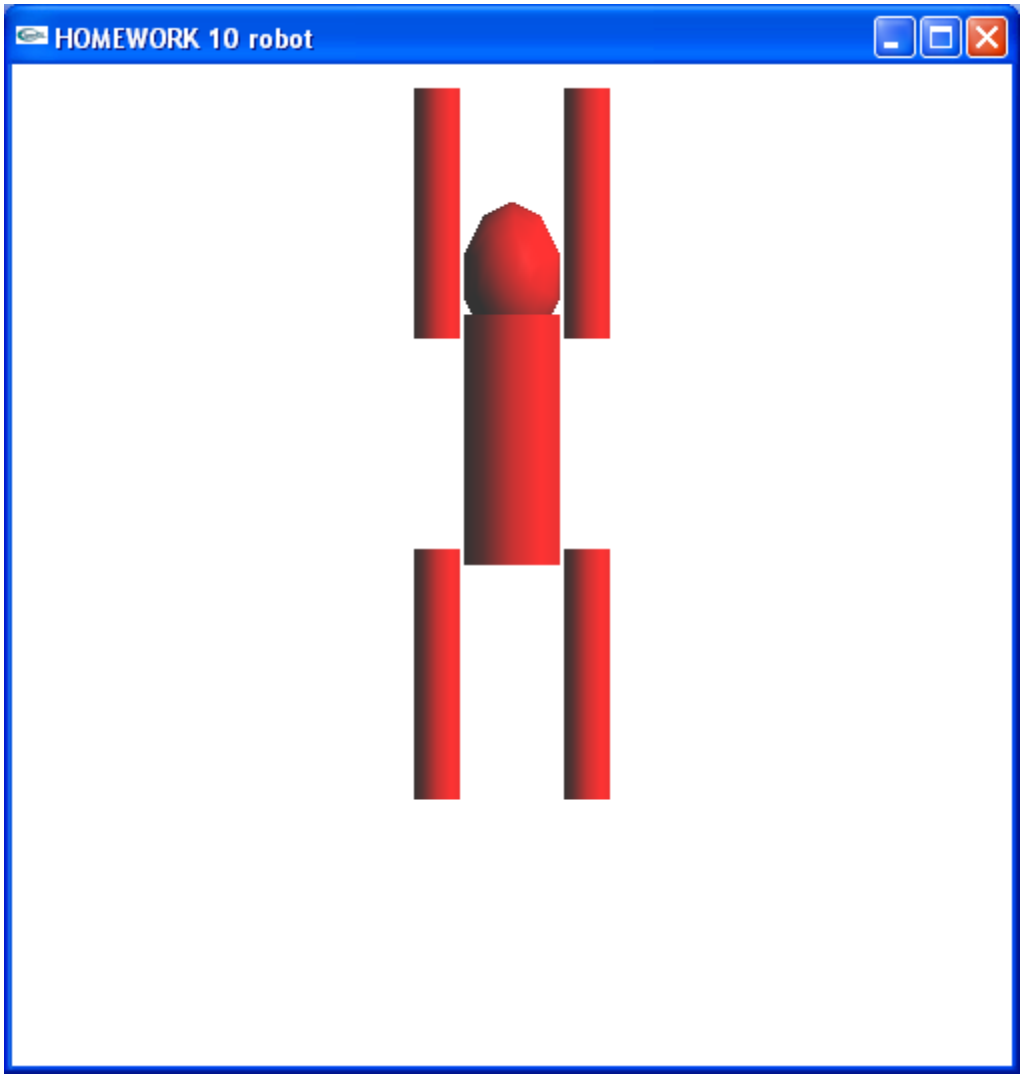
```
void left_lower_arm()
{
   glPushMatrix();
   glRotatef(-90.0, 1.0, 0.0, 0.0);
   gluCylinder(lla,LOWER_ARM_RADIUS, LOWER_ARM_RADIUS,
LOWER_ARM_HEIGHT,10,10);
   glPopMatrix();
}

void right_upper_arm()
{
   glPushMatrix();
   glRotatef(-90.0, 1.0, 0.0, 0.0);
   gluCylinder(rua,UPPER_ARM_RADIUS, UPPER_ARM_RADIUS,
UPPER_ARM_HEIGHT,10,10);
   glPopMatrix();
}

void right_lower_arm()
{
   glPushMatrix();
   glRotatef(-90.0, 1.0, 0.0, 0.0);
   gluCylinder(rla,LOWER_ARM_RADIUS, LOWER_ARM_RADIUS,
LOWER_ARM_HEIGHT,10,10);
   glPopMatrix();
}

void left_upper_leg()
{
   glPushMatrix();
   glRotatef(-90.0, 1.0, 0.0, 0.0);
   gluCylinder(lul,UPPER_LEG_RADIUS, UPPER_LEG_RADIUS,
UPPER_LEG_HEIGHT,10,10);
   glPopMatrix();
}

void left_lower_leg()
{
   glPushMatrix();
   glRotatef(-90.0, 1.0, 0.0, 0.0);
   gluCylinder(lll,LOWER_LEG_RADIUS, LOWER_LEG_RADIUS,
LOWER_LEG_HEIGHT,10,10);
   glPopMatrix();
}

void right_upper_leg()
{
   glPushMatrix();
   glRotatef(-90.0, 1.0, 0.0, 0.0);
   gluCylinder(rul,UPPER_LEG_RADIUS, UPPER_LEG_RADIUS,
UPPER_LEG_HEIGHT,10,10);
   glPopMatrix();
}

void right_lower_leg()
{
   glPushMatrix();
```

```
   glRotatef(-90.0, 1.0, 0.0, 0.0);
   gluCylinder(rll,LOWER_LEG_RADIUS, LOWER_LEG_RADIUS,
LOWER_LEG_HEIGHT,10,10);
   glPopMatrix();
}

void
display(void)
{
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glColor3f(1.0, 0.0, 0.0);

    glRotatef(theta[0], 0.0, 1.0, 0.0);
    torso();
    glPushMatrix();

    glTranslatef(0.0, TORSO_HEIGHT+0.5*HEAD_HEIGHT, 0.0);
    glRotatef(theta[1], 1.0, 0.0, 0.0);
    glRotatef(theta[2], 0.0, 1.0, 0.0);
    glTranslatef(0.0, -0.5*HEAD_HEIGHT, 0.0);
    head();

    glPopMatrix();
    glPushMatrix();
    glTranslatef(-(TORSO_RADIUS+UPPER_ARM_RADIUS), 0.9*TORSO_HEIGHT,
0.0);
    glRotatef(theta[3], 1.0, 0.0, 0.0);
    left_upper_arm();

    glTranslatef(0.0, UPPER_ARM_HEIGHT, 0.0);
    glRotatef(theta[4], 1.0, 0.0, 0.0);
    left_lower_arm();

    glPopMatrix();
    glPushMatrix();
    glTranslatef(TORSO_RADIUS+UPPER_ARM_RADIUS, 0.9*TORSO_HEIGHT, 0.0);
    glRotatef(theta[5], 1.0, 0.0, 0.0);
    right_upper_arm();

    glTranslatef(0.0, UPPER_ARM_HEIGHT, 0.0);
    glRotatef(theta[6], 1.0, 0.0, 0.0);
    right_lower_arm();

    glPopMatrix();
    glPushMatrix();
    glTranslatef(-(TORSO_RADIUS+UPPER_LEG_RADIUS),
0.1*UPPER_LEG_HEIGHT, 0.0);
    glRotatef(theta[7], 1.0, 0.0, 0.0);
    left_upper_leg();

    glTranslatef(0.0, UPPER_LEG_HEIGHT, 0.0);
    glRotatef(theta[8], 1.0, 0.0, 0.0);
    left_lower_leg();

    glPopMatrix();
    glPushMatrix();
```

```
    glTranslatef(TORSO_RADIUS+UPPER_LEG_RADIUS, 0.1*UPPER_LEG_HEIGHT,
0.0);
    glRotatef(theta[9], 1.0, 0.0, 0.0);
    right_upper_leg();

    glTranslatef(0.0, UPPER_LEG_HEIGHT, 0.0);
    glRotatef(theta[10], 1.0, 0.0, 0.0);
    right_lower_leg();

    glPopMatrix();
    glFlush();
    glutSwapBuffers();
}



void mouse(int btn, int state, int x, int y)
{
      if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN)
        {
        theta[angle] += 5.0;
        if( theta[angle] > 360.0 ) theta[angle] -= 360.0;
        }
      if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
        {
        theta[angle] -= 5.0;
        if( theta[angle] < 360.0 ) theta[angle] += 360.0;
        }
        display();
}

void menu(int id)
{
   if(id <11 ) angle=id;
   if(id ==11 ) exit(0);
}

void
myReshape(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w <= h)
        glOrtho(-10.0, 10.0, -10.0 * (GLfloat) h / (GLfloat) w,
            10.0 * (GLfloat) h / (GLfloat) w, -10.0, 10.0);
    else
        glOrtho(-10.0 * (GLfloat) w / (GLfloat) h,
            10.0 * (GLfloat) w / (GLfloat) h, 0.0, 10.0, -10.0, 10.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void myinit()
{
        GLfloat mat_specular[]={1.0, 1.0, 1.0, 1.0};
        GLfloat mat_diffuse[]={1.0, 1.0, 1.0, 1.0};
```

```
        GLfloat mat_ambient[]={1.0, 1.0, 1.0, 1.0};
        GLfloat mat_shininess={100.0};
        GLfloat light_ambient[]={0.0, 0.0, 0.0, 1.0};
        GLfloat light_diffuse[]={1.0, 0.0, 0.0, 1.0};
        GLfloat light_specular[]={1.0, 1.0, 1.0, 1.0};
        GLfloat light_position[]={10.0, 10.0, 10.0, 0.0};

        glLightfv(GL_LIGHT0, GL_POSITION, light_position);
        glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
        glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
        glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);

        glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
        glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
        glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess);

        glShadeModel(GL_SMOOTH);
        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
        glDepthFunc(GL_LEQUAL);
        glEnable(GL_DEPTH_TEST);

        glClearColor(1.0, 1.0, 1.0, 1.0);
        glColor3f(1.0, 0.0, 0.0);

/* allocate quadrics with filled drawing style */

        h=gluNewQuadric();
        gluQuadricDrawStyle(h, GLU_FILL);
        t=gluNewQuadric();
        gluQuadricDrawStyle(t, GLU_FILL);
        lua=gluNewQuadric();
        gluQuadricDrawStyle(lua, GLU_FILL);
        lla=gluNewQuadric();
        gluQuadricDrawStyle(lla, GLU_FILL);
        rua=gluNewQuadric();
        gluQuadricDrawStyle(rua, GLU_FILL);
        rla=gluNewQuadric();
        gluQuadricDrawStyle(rla, GLU_FILL);
        lul=gluNewQuadric();
        gluQuadricDrawStyle(lul, GLU_FILL);
        lll=gluNewQuadric();
        gluQuadricDrawStyle(lll, GLU_FILL);
        rul=gluNewQuadric();
        gluQuadricDrawStyle(rul, GLU_FILL);
        rll=gluNewQuadric();
        gluQuadricDrawStyle(rll, GLU_FILL);
}

void main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutCreateWindow("HOMEWORK 10 robot");
    myinit();
```

```
    glutReshapeFunc(myReshape);
    glutDisplayFunc(display);
    glutMouseFunc(mouse);

    glutCreateMenu(menu);
    glutAddMenuEntry("torso", 0);
    glutAddMenuEntry("head1", 1);
    glutAddMenuEntry("head2", 2);
    glutAddMenuEntry("right_upper_arm", 3);
    glutAddMenuEntry("right_lower_arm", 4);
    glutAddMenuEntry("left_upper_arm", 5);
    glutAddMenuEntry("left_lower_arm", 6);
    glutAddMenuEntry("right_upper_leg", 7);
    glutAddMenuEntry("right_lower_leg", 8);
    glutAddMenuEntry("left_upper_leg", 9);
    glutAddMenuEntry("left_lower_leg", 10);
    glutAddMenuEntry("quit", 11);
    glutAttachMenu(GLUT_MIDDLE_BUTTON);

    glutMainLoop();
}
```


HOMEWORK 10 robot